POCKET
COMPUTER
NEWSLETTER

(c) Copyright 1984 POCKET COMPUTER NEWSLETTER

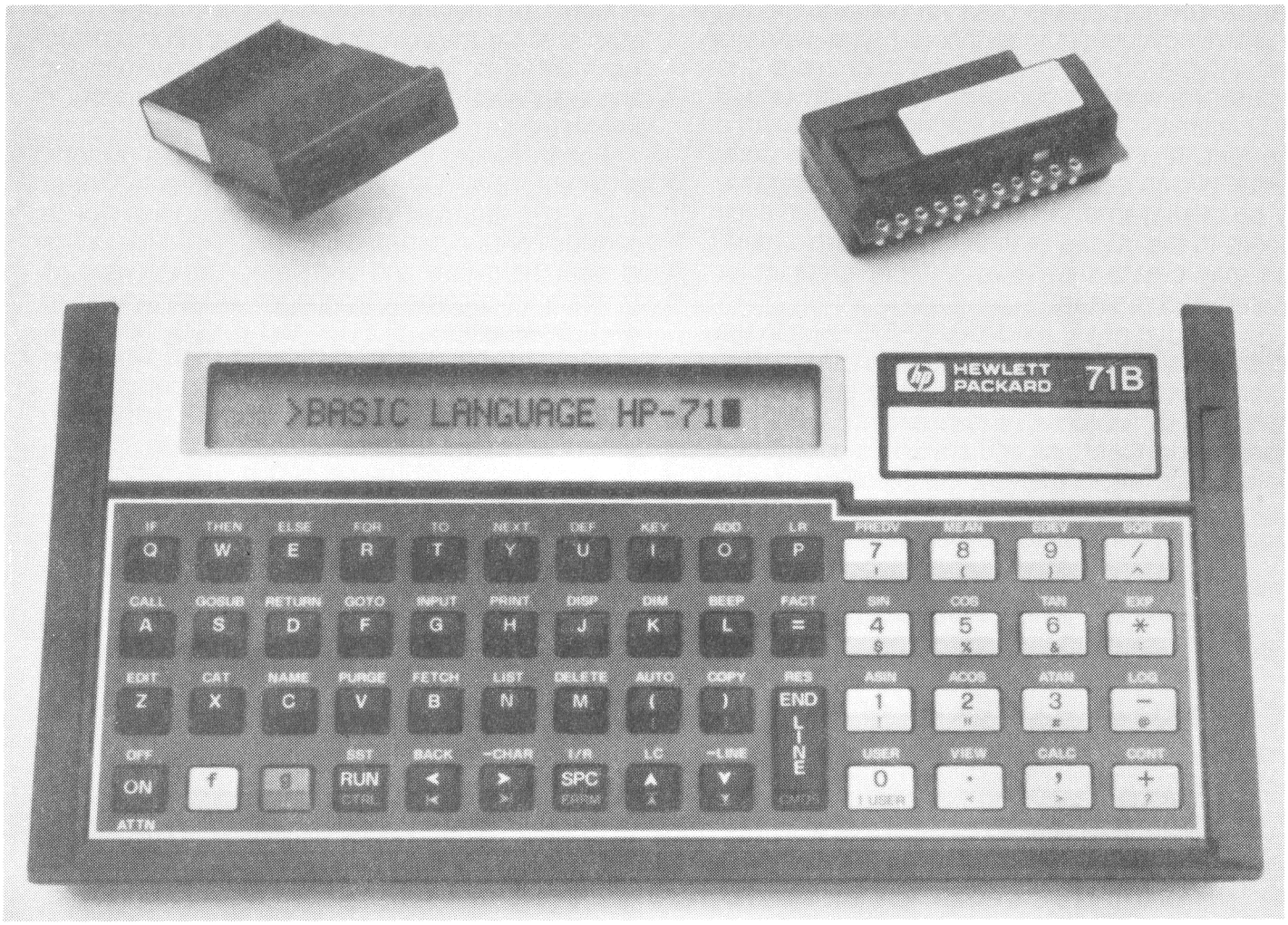
Issue 32 - March/April

NEW POCKET COMPUTER FROM HP

Hewlett-Packard Company has announced a new handheld computer dubbed the HP-71B. With measurements of just 3-7/8 by 7-1/2 by 1/2 inches and a weight of only 12 ounces, the unit fits easily into a jacket pocket. It was designed especially for technical professionals and has been optimized for numeric computations and calculations.

HP claims that it has the "most powerful 64-kilobyte ROM-based operating system available in a handheld computer." The machine combines an extended version of BASIC language

Photo *Hewlett-Packard HP-71B Shown With Optional HP-IL Module and Magnetic Strip-Card Reader.*



and a calculation mode to provide people with a highly portable tool capable of serious number crunching. Repetitive calculations may be performed by writing programs in BASIC or buying application packages that solve specific types of problems. A special CALC mode makes it easy to perform single calculations.

The 64 kilobytes of ROM is accompanied by 17.5 kilobytes of RAM in the basic unit. The unit is equipped with four slots that can accept additional RAM or ROM modules. Up to 16K of extra RAM or 256K of extra ROM can be added to the unit using these slots.

Space is also provided in the machine to install an optional magnetic card reader and an HP Interface Loop (HP-IL). The card reader reads (and writes) 10-inch magnetic strips that can hold up to 1.3 kilobytes of information. The HP-IL modules permits the HP-71B to be connected to a variety of peripheral devices such as a magnetic tape drive, printers and test equipment.

The pocketable unit is powered by four "AAA" alkaline batteries or by an optional alternating current adaptor.

The 71B has a block QWERTY-style keyboard with typing aids for facilitating use of BASIC. A separate 10-digit pad is used for numeric inputs. The entire keyboard is redefinable. Mylar overlays are available so that users can customize the computer for specific purposes. The display is an 8 by 132-element, single-line dot matrix LCD with a large font. It is supported by status annunciators. Twenty-two characters within a 96-character line may be displayed at a time. However, since each element in the display is individually controllable, users may create their own set(s) of graphics or customized characters.

The version of HP extended BASIC used in the machine provides over 240 instructions. In addition to compliance with IEEE floating-point math standards, it includes enhancements such as trigonometric functions and statistics operations. The language also provides: complete control of display pixels, ability to maintain multiple programs in memory, dynamically declared variables, and the ability to define multi-line user functions. Furthermore, additional keywords can be added to extend the language.

A built-in timer allows events to be controlled in real time. Time and date information is retained even when the HP-71B is turned off.

The handheld computer also has a calculation mode that enables it to function as an advanced calculator. Even though this mode is not programmable, it does provide error-checking and results can be tracked as a calculation is performed. Variables assigned values in the BASIC mode can be used in the CALC mode (as well as the reverse).

HP plans to support the new handheld with a series of application programs including math, curve fitting, AC circuit analysis, surveying, finance and text-editing. The firm also plans to provide a combination Assembly/FORTH System for developing special application programs. There will also be a user's library made available for the new model. Programs submitted by users will be made available through the library.

Hewlett-Packard Company has also indicated that the HP-71B will be a completely open machine. That is, it is encouraging third-party vendors to develop software, hardware and interfaces for the computer. It is making extensive documentation and a choice of development languages available as well as offering support of several media on which software may be delivered.

For instance, in addition to the usual owner's documentation, HP is making available three volumes of internal design specifications. These manuals contain detailed information on the unit's internal hardware and software. Three types of software development languages will be available: HP-71B Assembly, FORTH and BASIC. Software vendors will be able to distribute their wares on ROMs, cassette tapes and magnetic strip cards.

The HP-71B is available from HP authorized dealers. It is said to be priced at less than $550.00. For the location of the nearest dealer phone, toll free, 1-800-FOR-HPPC.

SCHEDULE PLOTTER PROGRAM

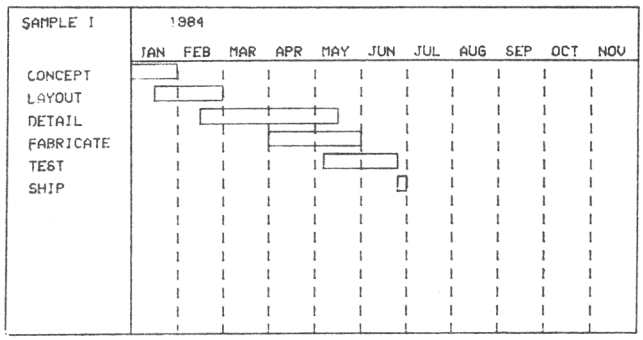
Richard H. Chrystie, 14824 E. Walbrook, Hacienda Heights, CA 91745, submitted this program. You can use it to plot schedules over a period of weeks or months.

To use it, just load the program and execute a RUN command or press the DEF key and and then the S key. Respond to the queries and watch your schedule be plotted right before your eyes!

You can customize the program to your own liking. For instance, if, when using it in the monthly mode, you want it to start with a month other than January, change the start date in line 620. You can then change the column month labels that begin at line 780.

Program *Schedule Plotter.*

```
 10:"S"REM   SCHED
    PLOT
 11:TEXT :COLOR 0:
    CLEAR
 15:DIM A$(12),B(1
    2),C(12)
 20:INPUT "PROGRAM
     TITLE?",B$
100:GOSUB 700
110:GLCURSOR (0,-4
    00)
120:IF Z$="M"THEN
    600
400:N=1
410:INPUT "TASK TI
    TLE?",A$(N)
420:INPUT "START (
    DAYS FROM NOW)
    ?",B(N):B(N)=B
    (N)*6
430:INPUT "TASK DU
    RATION (DAYS)?
    ",C(N):C(N)=C(
    N)*6
435:IF (B(N)+C(N))
    >330THEN LET C
    (N)=330-B(N)
440:L=183-(15*N)
449:COLOR 2:K=-80
450:GLCURSOR (L,-1
    3):LPRINT A$(N
    )
455:LINE (L,K-B(N)
    )-(L+10,K-B(N)
    -C(N)),,,B
460:GLCURSOR (0,-3
    80)
470:IF N=12THEN 49
    8
480:INPUT "WANT AN
    OTHER TASK?",D
    $
481:IF D$="N"THEN
    498
485:N=N+1
487:GOTO 410
498:GLCURSOR (0,-5
    00)
499:INPUT "WANT AN
    OTHER PLOT?";Z
    $
500:IF Z$="Y"THEN
    502
501:GLCURSOR (0,-5
    00):END
502:GOTO 11
600:N=1
610:INPUT "TASK TI
    TLE?",A$(N)
620:INPUT "START (
    MOS AFTER 1 JA
    N)?",B(N):B(N)
    =B(N)*30
630:INPUT "TASK DU
    RATION (MOS)?"
    ,C(N):C(N)=C(N
    )*30
635:IF (B(N)+C(N))
    >330THEN LET C
    (N)=330-B(N)
640:L=183-(15*N)
649:COLOR 2:K=-80
650:GLCURSOR (L,-1
    3):LPRINT A$(N
    )
655:LINE (L,K-B(N)
    )-(L+10,K-B(N)
    -C(N)),,,B
660:GLCURSOR (0,-3
    80)
670:IF N=12THEN 69
    8
680:INPUT "WANT AN
    OTHER TASK?",D
    $
681:IF D$="N"THEN
    698
685:N=N+1
687:GOTO 610
698:GLCURSOR (0,-5
    00)
699:GOTO 499
700:GRAPH
710:COLOR 0
715:LINE (0,0)-(21
    5,-410),,,B
720:LINE (0,-80)-(
    215,-80)
730:LINE (180,-80)
    -(180,-410)
740:FOR X=110TO 37
    0STEP 30
745:LINE (0,-X)-(1
    80,-X),5
750:NEXT X
755:CSIZE 1:ROTATE
    1:COLOR 1
760:GLCURSOR (203,
    -10)
775:LPRINT B$
776:INPUT "WEEKS O
    R MONTHS (W/M)
    ?",Z$
777:IF Z$="W"THEN
    800
780:GLCURSOR (183,
    -85):LPRINT "J
    AN"
781:GLCURSOR (183,
    -115):LPRINT "
    FEB"
782:GLCURSOR (183,
    -145):LPRINT "
    MAR"
783:GLCURSOR (183,
    -175):LPRINT "
    APR"
784:GLCURSOR (183,
    -205):LPRINT "
    MAY"
785:GLCURSOR (183,
    -235):LPRINT "
    JUN"
786:GLCURSOR (183,
    -265):LPRINT "
    JUL"
787:GLCURSOR (183,
    -295):LPRINT "
    AUG"
788:GLCURSOR (183,
    -325):LPRINT "
    SEP"
789:GLCURSOR (183,
    -355):LPRINT "
    OCT"
790:GLCURSOR (183,
    -385):LPRINT "
    NOV"
791:GLCURSOR (203,
    -105):LPRINT "
    1984"
792:RETURN
800:GLCURSOR (183,
    -90):LPRINT "1
    "
801:GLCURSOR (183,
    -120):LPRINT "
    2"
802:GLCURSOR (183,
    -150):LPRINT "
    3"
803:GLCURSOR (183,
    -180):LPRINT "
    4"
804:GLCURSOR (183,
    -210):LPRINT "
    5"
805:GLCURSOR (183,
    -240):LPRINT "
    6"
806:GLCURSOR (183,
    -270):LPRINT "
    7"
807:GLCURSOR (183,
    -300):LPRINT "
    8"
808:GLCURSOR (183,
    -330):LPRINT "
    9"
809:GLCURSOR (183,
    -360):LPRINT "
    10"
810:GLCURSOR (183,
    -390):LPRINT "
    11"
811:GLCURSOR (203,
    -105):LPRINT "
       WEEKS"
812:RETURN
STATUS 1        1722
```



APPOINTMENT CALENDAR

This program can be of use to almost everybody. It is particularly valuable because it gives you the capability of searching for a key word within a field to help you locate a specific appointment.

For example, suppose you had been invited to a golf game on Friday, September 23rd. But, you do not remember the exact time. Just run the program and select item 2 (Find Appointment) from the menu. Respond to the prompt for a date. If you make a mistake while entering it, error-trapping will advise you and prompt you again. Once you have entered the date, the PC will list all your committments on that date. The listing will be in the order in which you entered them. (In keeping with a goal of having enough memory available for a practical calendar, a rather lengthy sorting routine was omitted.) When the appointments for the given date have been shown, the program returns to its main memory.

Here is another example. Suppose you are running out of a favorite product that is sold only by Amway at Amway parties. You cannot remember when the next party is scheduled. Just run the program and select item 3 (Find Date) from the menu. In a few seconds the computer will begin displaying all of your Amway parties scheduled for the entire year! You could even spell Amway wrong and, unless you really mutilate it, the program will still find it for you!

If you like, you can also review appointments by the month (choose item 4 from the main menu). Other capabilities included in the program are: deleting the appointment file, saving data, loading data and adding appointments.

Operating Tips

There are a few things you need to remember about the PC-2 and PC-1500 Pocket Computers when

Program *Appointment Calendar.*

```
  5:DIM M$(60)*25,
    N$(60)*25,J$(2
    )*2,K$(2)*5
 10:"A":CLS :WAIT
    99:CURSOR 4:
    PRINT "APPOINT
    MENT CALENDAR"
 12:CLS :BEEP 1,10
    0
 20:CLS :CURSOR 8:
    PRINT "***MENU
    ***"
 22:CLS
 30:PAUSE "1.ENTER
     DATES/APPOINT
    MENTS"
 40:PAUSE "2.FIND
    APPOINTMENT"
 50:PAUSE "3.FIND
    DATE"
 60:PAUSE "4.REVIE
    W APPTS BY MON
    TH
 70:PAUSE "5.I/O O
    PTIONS     "
 75:PAUSE "6 DELET
    E APPOINTMENT"
 80:PAUSE "7.SEE M
    ENU AGAIN"
 90:INPUT M
100:ON MGOSUB 1000
    ,2000,3000,400
    0,5000,6000,70
    00
110:GOTO 12
 1000:BEEP 1,150:
      PRINT "IS TH
      IS AN ORIGIN
      AL ENTRY?"
 1002:INPUT "
           (Y/N)?
      ";J$
 1003:IF J$="N"
      THEN GOTO 80
      00
 1004:FOR L=1TO 60
 1005:INPUT "ENTER
       DATE (MO/DA
      ): ";M$(L)
 1015:IF M$(L)="ZZ
      "THEN GOTO 1
      050
 1016:IF LEN (M$(L
      ))<>5THEN
      GOTO 9000
 1020:INPUT "ENTER
       APPOINTMENT
      :";N$(L)
 1030:NEXT L
 1040:GOTO 20
 1050:RETURN
 2000:BEEP 1,150
 2010:PAUSE "FIND
      APPOINTMENT"
 2015:INPUT "ENTER
       DATE (MO/DA
      ): ";K$
 2020:IF LEN (K$)<
      >5THEN BEEP
      5,150,25:
      PRINT "FORMA
      T IS 4 DIGIT
      S---REDO":
      GOTO 2015
 2021:X=1:Y=0:GOTO
      2025
 2024:X=X+1
 2025:FOR X=XTO L-
      1
 2030:IF K$=M$(X)
      THEN 2070
 2040:NEXT X
 2045:IF Y>=1THEN
      2110
 2050:PRINT "NO AP
      POINTMENT ON
      : ";K$
 2055:K$=""
 2060:RETURN
 2070:PRINT "YOUR
      APPOINTMENT
      ON: ";M$(X);
      "IS:"
 2075:PRINT "IS: "
      ;N$(X)
 2077:Y=Y+1
 2080:FOR C=1TO 75
      :NEXT C
 2093:GOTO 2024
 2110:PRINT "NO MO
      RE APPTS ON
      THIS DATE"
 2120:K$=""
 2130:RETURN
 3000:BEEP 1,150
 3005:CLS :CURSOR
      8:PRINT "FIN
      D DATE"
 3010:INPUT "ENTER
       APPOINTMENT
      ";U$
 3012:X=0:Y=0
 3014:X=X+1
 3020:FOR X=XTO L-
      1
 3025:IF U$=N$(X)
      THEN 3050
 3030:IF LEFT$ (N$
      (X),4)=LEFT$
      (U$,4)THEN 3
      050
 3035:IF RIGHT$ (N
      $(X),4)=
      RIGHT$ (U$,4
      )THEN 3050
 3040:NEXT X
 3042:IF Y>=1THEN
      3090
 3045:GOTO 3080
 3050:PRINT M$(X);
      " ";N$(X)
 3052:Y=Y+1
 3055:FOR R=1TO 10
      0:NEXT R
 3057:GOTO 3014
 3060:U$=""
 3070:RETURN
 3080:PRINT "THAT
      APPT IS NOT
      ON FILE"
 3090:U$=""
 3100:RETURN
 4000:CLS :BEEP 1,
      150
```

using this program.

First, remember that when you give a RUN command, all variables stored in user memory will be lost. Thus, you should only use the RUN command when you use the program for the first time or after loading the *program* from tape. It is necessary to do so under these circumstances in order to properly dimension the string arrays used by the program.

How do you start the program without typing RUN? Just press the DEF key followed by the letter A (DEF/A).

In order to start building up an appointments list you will need to choose item 1 from the main menu. When you do this you will queried as to whether this is an original entry. If it is an original entry, (the first time you have entered information into an appointment file) respond with a Y for yes. A yes response causes the data file to be initialized to the empty condition. If you are adding data to a file that already has appointments stored, then be sure to answer N for no!

You can customize the display timings to your own liking. There are three basic methods of controlling the length of time a display stays on the screen of a PC-2 or PC-1500. The PAUSE and PRINT commands are used in this program. PAUSE gives a display time that is set at about one second. Using the PRINT command coupled with WAIT (followed by a number), you can keep the display on for whatever period is desired. A WAIT value of 99 is used in this program. It is specified in line 10. If you would like the displays to last a little longer, increase the value in line 10. Remember that altering this value will change all PRINT statements used in the program but has no effect on the PAUSE statements.

This program has been designed to fit in a 4K

```
4005:INPUT "ENTER
     MONTH TO RE
     VIEW ";H$
4008:IF LEN (H$)<
     >2THEN BEEP
     10,150,25:
     PRINT "FORMA
     T IS 2 DIGIT
     S---REDO":
     GOTO 4005
4020:X=0:Y=0
4030:X=X+1
4035:FOR X=XTO L-
     1
4040:IF H$=LEFT$
     (M$(X),2)
     THEN 4070
4050:NEXT X
4055:IF Y>=1THEN
     4110
4060:PRINT "NO AP
     PTS LISTED T
     HIS MONTH"
4065:H$="":RETURN
4070:PRINT RIGHT$
     (M$(X),2);"
     ";N$(X)
4080:Y=Y+1
4090:FOR C=1TO 75
     :NEXT C
4100:GOTO 4030
4110:PRINT "NO MO
     RE APPTS THI
     S MONTH"
4120:H$="":RETURN
5000:CLS :BEEP 1,
     150
5005:CLS :CURSOR
     4:PRINT "MEN
     U--I/O OPTIO
     NS"
5010:PAUSE "1.SAV
     E TO TAPE"
5020:PAUSE "2.LOA
     D FROM TAPE
     "
5030:PAUSE "3.RET
     URN TO MAIN
     MENU"
5033:PAUSE "4.SEE
      I/O MENU AG
     AIN"
5035:INPUT N
5040:ON NGOSUB 10
     000,11000,12
     000
5060:GOTO 5000
6000:CLS :BEEP 1,
     150:J=1
6005:CURSOR 5:
     PRINT "DELET
     E APPOINTMEN
     T"
6010:INPUT "APPOI
     NTMENT DATE
     (MO/DA) ";G$
6015:IF LEN (G$)<
     >5THEN BEEP
     5,150,25:
     PRINT "FORMA
     T IS 5 DIGIT
     S---REDO":
     GOTO 6010
6030:FOR X=1TO L-
     1
6040:IF G$=M$(X)
     THEN PRINT "
     ITEM #";X;"
      ";N$(X):J=J
     +1
6050:NEXT X
6051:IF J<=1THEN
     PRINT "THERE
     S NO APPT ON
      ";G$:GOTO 6
     100
6052:PRINT "SEE A
     PPTS ON ";G$
     ;" AGAIN?
6053:CLS :CURSOR
     9:PRINT "(Y/
     N)?"
6054:INPUT L$
6055:IF L$="Y"
     THEN 6030
6060:INPUT "DELET
     E ITEM NUMBE
     R: ";F
6070:N$(F)="":M$(
     F)=""
6075:L$=""
6080:RETURN
6100:G$="":RETURN
7000:GOTO 20
8000:CLS :BEEP 1,
     150
8005:PRINT "MAKE
     ADDITIONS TO
      CALENDAR
8010:FOR L=LTO 60
8030:GOTO 1005
9000:CLS :BEEP 10
     ,100,25
9005:PRINT "  FOR
     MAT IS (MO/D
     A) REDO
9010:GOTO 1005
10000:PRINT "SAVE
     TO TAPE"
10020:PRINT "PREPA
     RE TAPE RECO
     RDER"
10030:FOR B=1TO 20
     0:NEXT B
10040:PRINT #L,M$(
     *),N$(*)
10050:CLS :BEEP 1,
     150
10060:RETURN
11000:PRINT "LOAD
     FROM TAPE"
11010:PRINT "PREPA
     RE TAPE RECO
     RDER"
11020:FOR D=1TO 20
     0:NEXT D
11030:INPUT #L,M$(
     *),N$(*)
11040:CLS :BEEP 1,
     150
11050:RETURN
12000:GOTO 20
STATUS 1    2689
```

machine. If you have an 8K memory module installed (which nets you 10K), you can change the number of potential records by the changing the values in the loops at lines 1004 and 8010, to something in the order of 100. If you do this you must also change the numbers within the parenthesis of the dimension declarations (in line 5) to 100 instead of 60. If you assign too many records in the dimension statements, you will simply get an OUT OF MEMORY error when you try to use the program. Reduce the values used if this occurs.

Finally, if you have never saved a data file before using your PC, then you may be in for a bit of a surprise. Did you realize that even if you only have two appointment entries in your appointment file, a tape data save command will save all of the records that have been dimensioned, even though most of them are empty? Yes, the PC will save every bit of the reserved array space, whether or not it contains information. The result is that saving the appointment file on tape will take a fixed amount of time, regardless of how many entries you have made. Have patience, please.

This program was submitted by: *Steve Eichman, 5809 Northland Road, Manteca, CA 95336.*

FORMATTED MORTGAGE PROGRAM

Here is a program that will calculate monthly payments and amortization information on a mortgage or loan and print out a neatly formatted schedule.

The program asks for the mortgage/loan amount, annual interest percentage and amount of monthly payments. If the monthly payment is specified, the amortization will be calculated. Optionally, if the monthly payment is not given (simply press ENTER at the prompt), the program will ask the user for amortization and calculate the appropriate monthly payments. Next, the starting date and number of payments to be printed in the schedule must be provided. Finally, the program gives the option of specifying a number of extra payments. If only regular payments will be made, press the ENTER key at this prompt. When executed the program lists a summary of the input data, followed by a payment schedule that shows the payment number, date, interest and principal portions of the payment, and the account balance as of that date.

NOTICE: This program was written to calculate mortgage schedules according to Canadian laws. However, with a few simple changes it can be adapted to other types of loans.

In Canada, the law states that interest on mortgages may not be charged in advance. This means that a lender is not entitled to interest more often than the mortgage's compounding frequency which is a maximum of semi-annually. Thus, if interest payments are in fact being made more often than the law permits (such as monthly), then interest is being charged in advance, unless an appropriate discount is made. This may be done by reinvesting the interest portion of the monthly payment at the mortgage rate until the end of the current compounding period. This effectively reduces the actual interest rate. The effective

Program *Formatted Mortgage.*

```
  5:PAUSE "... MORTGAGE SCHEDULE ...":LOCK
 10:REM      1983 - Peter V. HARTMANN
 20:CLEAR :CSIZE 2:B=10:U=1:DIM PN(B),EP(B)
 30:A$="Jan":B$="Feb":C$="Mar":D$="Apr":E$="May":F$="Jun"
 40:G$="Jul":H$="Aug":I$="Sep":J$="Oct":K$="Nov":L$="Dec"
 50:INPUT "AMOUNT = $ ";A
 60:INPUT "INTEREST (%) = ";I
 70:C=2
 80:INPUT "MONTHLY PAYMENTS = $";P
 90:IF P=0INPUT "AMORTIZED OVER (yrs) = ";Z:N=Z*12
100:INPUT "STARTING DATE- DAY: ";D,"MONTH: ";M,"YEAR: ";Y
110:Y=Y-100*INT (Y/100)
120:INPUT "NO. OF PAYMENTS = ";U
130:INPUT "EXTRA PAYMENTS ? ";Z$
140:IF Z$<>"Y"GOTO 170
150:FOR J=0TO B:INPUT "PAYMENT # = ";PN(J)
155:IF PN(J)=0GOTO 170
160:INPUT "PAYMENT AMOUNT = $";EP(J):NEXT J
170:R=(1+(I/100)/C)^(C/12)-1
180:IF P>0GOTO 200
190:X=(R+1)^N:P=A*R*X/(X-1):GOTO 210
200:N=LOG (P/(P-A*R))/LOG (1+R):Z=N/12
210:COLOR 2:LPRINT "----------"
220:COLOR 3:LPRINT "MORTGAGE SCHEDULE "
230:COLOR 2:LPRINT "----------"
240:COLOR 0:LF 1:USING "########"
250:LPRINT "AMOUNT = $";A
260:LPRINT "INTEREST=";USING "###.##";I;"%"
270:LPRINT "YEARS =";USING "####.##";Z
```

monthly interest rate is given by the equation:

$$r = ((1 + i/c)^{(c/12)}) - 1$$

In this formula, r = the effective monthly interest rate, i = the annual interest rate and c = the number of compounding periods per year.

Program Options

The maximum number of extra payments that can be handled by the program is a function of the amount of memory available. The value of variable B in line 20 of the listing sets the allowable limit. In an unexpanded PC-1500 the number of such payments is limited to 11. (B is set to 10 and is then used to DIMension an array having 11 elements.) If you have additional memory in your PC you can increase this value accordingly.

```
280:COLOR 1:LF 1:
    LPRINT "EFF.MO
    NTHLY RATE:"
290:LPRINT USING "
    ##.############
    ";R;"%"
300:COLOR 3:LF 1:
    LPRINT "MONTHL
    Y PAYMENTS:"
310:LPRINT "    $"
    ;USING "#####.
    ##";P
320:COLOR 2:LPRINT
    "-------------
    -----"
330:LF 2:CSIZE 1:
    COLOR 0:IF V=0
    GOTO 510
340:LPRINT "  #
    DATE      INT.
      PRINC. BALAN
    CE"
350:LPRINT "------
    --------------
    --------------
    --"
360:LPRINT TAB 3;
    USING "###";D;
    @$(M);Y;TAB 29
    ;USING "######
    #";A
370:K=M+1:IF K>12
    LET Y=Y+1:IF K
    >12LET K=1
380:S=0:T=0
390:FOR J=KTO 12:Q
    =P
400:FOR F=1TO B:IF
    PN(F)<>UGOTO 4
    20
410:Q=EP(F):GOTO 4
    30
420:NEXT F
430:G=A*R:H=Q-G:A=
    A-H:IF A>0.01
    GOTO 450
440:H=H+A:Q=H+G:A=
    0
450:LPRINT USING "
    ###";U;D;@$(J)
    ;Y;USING "####
    ##.##";G;USING
    "######.##";H;
    USING "#######
    ";A
460:S=S+G:T=T+H:U=
    U+1:IF A=0GOTO
    480
470:NEXT J
480:LPRINT "------
    --------------
    --------------
    --";X=Y+1900
490:LPRINT " TOTAL
    ";USING "#####
    ";X;USING "###
    ####.##";S;
    USING "######.#
    #";T:LF 1
500:K=1:Y=Y+1:IF U
    <UAND A>0.01
    GOTO 380
510:UNLOCK :END
STATUS 1       1649
```

To use the program to calculate other types of mortgages or simple loans, change the title in line 5 to suit your purposes. Also change line 70 to:

70 INPUT COMPOUNDED(* A YEAR)";C

and alter line 270 to read:

270 R = I

This program was submitted by: *Peter V. Hartmann, P.O. Box 7003, Station A, Toronto, Ontario, Canada M5W 1X7.*

Sample *Listing from Formatted Mortage Program.*

```
------------------
MORTGAGE SCHEDULE
------------------

AMOUNT   = $  60000
INTEREST=    12.75%
YEARS    =    25.00

EFF.MONTHLY RATE:
 0.01035329508%

MONTHLY PAYMENTS:
     $  650.81
------------------
```

#	DATE	INT.	PRINC.	BALANCE
	17Aug 83			60000
1	17Sep 83	621.19	29.61	59970
2	17Oct 83	620.89	29.91	59940
3	17Nov 83	620.58	30.22	59910
4	17Dec 83	620.26	30.54	59879
TOTAL 1983		2482.93	120.30	
5	17Jan 84	619.95	30.85	59848
6	17Feb 84	619.63	31.17	59817
7	17Mar 84	619.30	31.50	59786
8	17Apr 84	618.98	31.82	59754
9	17May 84	618.65	32.15	59722
10	17Jun 84	618.32	32.48	59689
11	17Jul 84	617.98	32.82	59656
12	17Aug 84	617.64	33.16	59623
13	17Sep 84	617.30	33.50	59590
14	17Oct 84	616.95	33.85	59556
15	17Nov 84	616.60	34.20	59522
16	17Dec 84	616.25	34.56	59487
TOTAL 1984		7417.59	392.12	
17	17Jan 85	615.89	34.91	59452
18	17Feb 85	615.53	35.27	59417
19	17Mar 85	615.16	35.64	59381
20	17Apr 85	614.79	36.01	59345
21	17May 85	614.42	36.38	59309
22	17Jun 85	614.04	36.76	59272
23	17Jul 85	613.66	37.14	59235
24	17Aug 85	613.28	37.52	59197
25	17Sep 85	612.89	37.91	59159
26	17Oct 85	612.50	38.30	59121
27	17Nov 85	612.10	38.70	59082
28	17Dec 85	611.70	39.10	59043
TOTAL 1985		7366.00	443.71	
29	17Jan 86	611.29	39.51	59004
30	17Feb 86	610.88	39.92	58964
31	17Mar 86	610.47	40.33	58924
32	17Apr 86	610.05	40.75	58883
33	17May 86	609.63	41.17	58842
34	17Jun 86	609.21	41.60	58800
35	17Jul 86	608.77	42.03	58758
36	17Aug 86	608.34	42.46	58716
37	17Sep 86	607.90	42.90	58673
38	17Oct 86	607.46	43.34	58629
39	17Nov 86	607.01	43.79	58586
40	17Dec 86	606.55	44.25	58541
TOTAL 1986		7307.62	502.09	

FLIGHT COMPUTER USES PC-1500

A new flight computer dubbed the PETREL has been announced. The unit is based on a Sharp PC-1500 Pocket Computer. A custom made memory expansion module about the size of a matchbook expands the memory of the PC-1500 by 32K of ROM and 6K of RAM. This additional memory allows the unit to hold a data base containing the location of all 1000 VOR stations in the United States. This data base coupled with proprietory operating programs enables the unit to compute the latitude and longitude of any Victor airways intersection after entry of just the VOR identifiers and radials. The system can also compute as many RNAV waypoints as desired for great circle routes up to 5000 miles in length. Up to 104 intersections or waypoints may be stored by the user and recalled by identifier alone. A user can also add up to 127 new entries or changes to the VOR data base. Furthermore, a variety of flight planning, preflight, navigation and inflight functions are included in the computing package.

For instance, the preflight capabilities allow the pilot to enter the empty weight and moments for fuel, oil, seats and baggage just once. This information is retained by the system for future use. Before each flight the operator adds the actual fuel and weight at each station. The PETREL system then computes the total weight and center of gravity figures as well as maximum gross weight and C.G. limits. In fact, the package will store the data needed for two separate aircraft so that one craft can be flown without losing the data used by the other. Additionally, the unit handles all typical conversions used by pilots: miles-nautical to miles-kilometers-feet, farenheit-celsius, liters-gallons-imperial gallons, and kilograms-pounds. It also handles conversions of true airspeed from calibrated airspeed or Mach numbers. The system provides 10 auxiliary memory scratch pads so that converted numbers can be entered directly into operating programs.

While navigating the PETREL can compute the total distance, flight time and fuel requirements without map reference. Three separate flight plans can be stored and run independently. Winds and climb requirements are computed automatically. The distance and bearing between any two airway points or geographic points can be determined in seconds without map reference. ETAs and fuel requirements can then be computed. Using its self-contained VOR data base, LORAN-C and other users can compute and display in a matter of seconds, the latitude and longitude of any VOR intersection. Only the VOR identifier and radial is entered to obtain this information. For a DME fix, just add the distance. Up to 104 intersections or other fixes can be stored and later recalled using just the identifier.

While inflight the system aids pilots through the maintenance of two 15-item general purpose check lists. A holding pattern routine displays the proper pattern entry procedure and provides a countdown timer for the outbound leg. If you input the final approach course, distance and wind, the PETREL will compute the time to missed approach and display a countdown/countup timer with an alarm. The programs are designed to keep inflight key punching to a minimum. For instance, the Distance-Time-Fuel program allows the operator to enter just the distance, leaving the speed and fuel consumption as they had been previously defined, and quickly obtain a current read out of the flight time, fuel used and ETA.

The PETREL Flight Computer is the result of several years of programming and engineering development work. It represents one of the most powerful applications of a pocket computer ever announced. It uses a combination of machine language and BASIC routines plus a comprehensive VOR data base stored on ROM. The PETREL currently is priced at $695.00. For more information contact: *Somerset Engineering Company, P.O. Box 14, Concord, MA 01742.* The phone number is (617) 369-1090.

GRAPHICS PACKAGE FOR THE PC-2

Radio Shack has announced the availability of a program called called the PC-2 Graphics Pak. It is said that the software package makes it easy to create charts using the Radio Shack TRS-80 Pocket Computer Model PC-2 combined with the PC-2 Printer/Cassette Interface.

The package of programs can be used to draw six different types of charts: horizontal and vertical bar charts, bar-segment charts, point, line and pie charts.

Data may be entered from the keyboard of the PC. It is turned into chart format using combinations of labeling and shading. A data file used in a chart can be stored on cassette tape and used to create another chart.

The package retails for $19.95 in the U.S. at Radio Shack Computer Centers and other Radio Shack outlets.

INSIDE THE PC-1500

This is a continuation of the series by *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* We pick up in this issue from where we left off in *PCN* Issue 31, with a recap of information relating to the use of RAM in the PC-1500. (This information also applies to the Radio Shack PC-2.)

Then Norlin goes into the subject of ROM routines used in the PC-1500 (which are again generally applicable to the PC-2). There are many routines within the BASIC ROM that may be of use to machine language programmers. Norlin has done an amazing job of uncovering these routines and providing a synopsis of their capabilities. Space does not permit the publication of all the ROM information he has dug up in this issue, so look for more in future editions of *PCN*.

THE BASIC STACK

The BASIC stack, located 7A38-7AFF, includes a FOR stack, a GOSUB stack, and an arithmetic stack.

The FOR loop stack begins at 7A38. Up to 16 loops may be pending at a given time (if the BASIC stack is not required for other purposes). Each FOR register contains 12 bytes, used as follows:

0-1 Memory address of control variable
2-3 Test value (in signed binary)
4-5 Step size (in signed binary)
6-7 Address of next statement following FOR; plus 80 if not the first statement in a line.
8-9 Number of line containing the statement following FOR
A-B Address of beginning of program in progress

GOSUB return addresses are stacked beginning at 7AFF, proceeding in reverse direction through memory. Up to 33 return addresses may be pending (if the BASIC stack is not required for other purposes). Each GOSUB register contains 6 bytes, as follows:

0-1 Address of statement following GOSUB; plus 80 if not the first statement in a line.
2-3 Number of line containing the statement following GOSUB
4-5 Address of beginning of program in progress

The space between the FOR and GOSUB stacks comprises the arithmetic stack, which is used by the parsing routine in evaluating expressions. Pending numeric data (or string pointers) will be stacked (in 8-byte blocks) starting at 7A3C, or at the location four bytes after the last address used for storing a FOR register. Codes representing pending operations are stacked in reverse direction through memory, starting at 7AFF, or at the location just preceding the stacked GOSUB registers.

Each pending operation code contains two bytes, of which the first indicates the priority of the operation. Those representing functions consist of the token codes. (The tokens for SQR and PI represent the corresponding symbols.) The other pending operation codes follow:

84	5E	↑	70	51	OR
83	21	sign change	70	50	AND
82	2F	/	60	2C	comma separating arguments
82	2A	*			
81	2D	-	5A	xx	variable name

```
81 2B  +                        59 xx  variable name
80 06  >=                       ...
80 05  <=                       41 xx  variable name
80 04  =
80 02  >                        40 80  @(
80 01  <                        20 28  left-Parenthesis
80 00  <>
```

SYSTEM RAM

7B00-7B07 contain the current value of the random number.

7B08 Unused
7B09 Counter for time that a key is held down
7B0A High byte, timed Powerdown counter (counts from FE1D1D to FFFFFF)
7B0B Second byte, timed Powerdown counter
7B0C Low byte, timed Powerdown counter
7B0D Cursor Blink counter (counts from 80 to FF)
7B0E Cursor Control Parameter: 80 indicates down arrow key required to continue Program execution. Other bits control cursor repeat.
7B0F Key matrix code for depressed key

STRING BUFFER

The 80-byte String Buffer, located 7B10-7B5F, contains the codes for character strings formed by the Parsing of string expressions.

The String Buffer and Output Buffer together form a temporary storage area into which the Previous contents of the Display Buffer are saved, during display of a Program line at an execution halt, or during the display of a BREAK message or RESERVE template.

Certain Printing routines use Portions of the String Buffer as a scratch area. Address 7B1F is used as a flag, containing 00 for LLIST and FF for auto Print of the Input Buffer contents in immediate mode.

OUTPUT BUFFER

The 80-byte Output Buffer is located 7B60-7BAF.

The codes for the characters to be displayed in response to a PRINT, PAUSE, or immediate calculation, are stored, in the required format, in addresses 7B60-7B79. (Codes for numeric output are first Placed into the 7A10-34 area, then transferred here to construct the display; codes for string output are transferred from the string buffer.)

With LPRINT, the characters representing numeric data to be Printed are stored, in the required format, in the area 7B80-7B9E. Another Portion of the Output Buffer (ending 7B7F) serves as a scratch area used to form Printed characters. Other locations are used as follows:

7BA9 A line feed (following Printing) is specified by 04
7BAA USING editing character: 10, comma separation; 20, forced sign; 40, asterisk fill; 80, scientific
7BAB USING number of characters Preceding decimal Point
7BAC USING number of characters in string; 00 if unspecified
7BAD USING number of characters including and following decimal Point
7BAE High byte, saved value of register Y
7BAF Low byte, saved value of register Y

Graphic Printing routines use the area 7B92-7BAD to store coordinates of points specified by LINE or RLINE statements, as well as of the present location. The following addresses are also used:

7B83 Flag: 00, Line; FF, Box
7B84 Count of number of line segments to be drawn
7B85 Flag: 00, LINE; nonzero, RLINE

Prior to recording (or locating) a cassete file, a header is formed in the Output Buffer, located as follows:

7B60-67 Lead-in; consists of the bytes 10, 11, ..., 19
7B68 File type: 00, ML; 01, BASIC; 02, RESERVE; 04, data
7B69-78 File name; null bytes if unspecified
7B79-81 Unused
7B82-83 Beginning memory address of file; not used for data files
7B84-85 Number of bytes in file, less 1; zero for data file
7B86-87 Starting address for automatic execution, ML files only; if not specified, FFFF.

During a load operation, header information read from the recorded file is stored into the Output buffer as follows:

7A91-A0 File name
7AA1-A9 Unused
7AAA-AB Beginning memory address (used when CLOAD M statement does not specify an address)
7AAC-AD Number of bytes in file, less 1; zero for data file
7AAE-AF Starting address for automatic execution of ML Program

INPUT BUFFER

The 80-byte Input Buffer, located 7BB0-7BFF, holds the codes for characters as they are entered from the keyboard. It is also used for storing codes that form input prompts, displayed Program lines, expressions to be evaluated in immediate mode, and displayed RESERVE key phrases.

USABLE ROM SUBROUTINES

The list which follows contains information related to a number of subroutines in ROM that are usable in user machine-language Programs.

A subroutine that is addressable as a Vector Call will be listed as 'Call nn'; the address of the subroutine will follow, in Parentheses.

CLEARING AND MOVING

BLOCKS OF MEMORY

Call BA (F763) clears the contents of memory from addresses X to X+UL. Register Y is not affected.

Subroutine D3C5 clears memory from addresses X to X+U; Register Y is not affected.

Subroutine D3C7 fills memory, from addresses X to X+U, with the byte contained in Register A. Register Y is not affected.

CLEARING OF BUFFERS

Call F2 (EE71) clears the Display Buffer (7600-764D and 7700-774D). Register Y is not affected.

Subroutine EF81 clears the Output Buffer (7B60-7BAF). Register Y is not affected.

Subroutine D02B fills the Input Buffer with 0D codes. Both Register Y and the Input Buffer Pointer (788B) are set to point to 7BB0, the beginning address of the Input Buffer.

DATA TRANSFERS TO BUFFERS

Call 94 (EC5C) copies the contents of addresses X to X+UL-1 into the Output Buffer, starting at the position specified by the Output Buffer Pointer (788F). If insufficient room is available, flag C will be set.

Subroutine EDC1 copies the contents of the Display Buffer into the area containing the String and Output Buffers. Subroutine EDD8 copies these codes back into the Display Buffer.

Subroutine FBCB, with C clear, copies the contents of the String Buffer into the Output Buffer.

Subroutine FBCB, with C set, copies the contents of the Output Buffer into the String Buffer.

FLOATING-POINT ARITHMETIC

The floating-point routines use the BASIC registers located at the following addresses:

R0: 7A00-7A07
R2: 7A10-7A17
R6: 7A30-7A37

Register R0 act as a floating-point accumulator, with R2 containing the second operand. Register R6 is used for temporary storage; except as noted, it is unaffected by the floating-point arithmetic routines.

The 8 bytes in a floating-point register are used as follows:

Byte 0	Exponent (in signed binary)
Byte 1	Sign: 00 = +, 80 = -
Byte 3	1st and 2nd mantissa digits (binary-coded decimal)
Byte 4	3rd and 4th mantissa digits
Byte 5	5th and 6th mantissa digits
Byte 6	7th and 8th mantissa digits
Byte 7	9th and 10th mantissa digits
Byte 8	11th and 12th mantissa digits

PRELOADING OF XH AND YH

In the case of certain indicated routines, XH and YH must contain 7A prior to execution of the routine.

Call 54 (F7B0) will load XH and YH with 7A.

Unless otherwise noted, both XH and YH will contain 7A following execution of any of the floating-point arithmetic routines listed.

DATA TRANSFER BETWEEN R0 AND FIXED VARIABLES

Variable into R0: subroutine DC20. Precede by loading X with initial address of variable. (Following execution, XH and YH will not necessarily contain 7A.)
R0 into variable: subroutine DC0C. Precede by loading X with initial address of variable. Register Y will be unchanged by this routine.

The initial addresses of fixed numeric variables are as follows:

A 7900	E 7920	I 7940	N 7968	R 7988	V 79A8
B 7908	F 7928	J 7948	O 7970	S 7990	W 79B0
C 7910	G 7930	K 7950	P 7978	T 7998	X 79B8
D 7918	H 7938	L 7958	Q 7980	U 79A0	Y 79C0
		M 7960			Z 79C8

CLEARING OF REGISTERS

Clear R0: Call EC (F757). Register Y is unchanged by this routine.
Clear R2: Subroutine F753. (XH and YH must contain 7A.)

DATA TRANSFER BETWEEN REGISTERS

R0 into R2: Call E6 (F70D).
R0 into R6: Call 80 (F707).

R2 into R0: Call 56 (F73D). (XH and YH must contain 7A.)
R2 into R6: Subroutine F701. (XH and YH must contain 7A.)

R6 into R0: Subroutine F737.
R6 into R2: Call 68 (F715). (XH and YH must contain 7A.)

Exchange R0, R2: Call 66 (F7B9). (XH and YH must contain 7A.)
Exchange R0, R6: Call 64 (F7B5). (XH and YH must contain 7A.)

DATA TRANSFER BETWEEN R0 AND BASIC STACK

The BASIC stack, located 7A3C-7AFF, permits up to 24 PUSHes of R0.

PUSH R0 onto BASIC stack: Subroutine DBF5.
POP R0 from BASIC stack: Call 30 (DC16)

ARITHMETIC OPERATIONS

All calculated results are normalized, with underflow to zero; each contains a 12-digit mantissa. Calculation of trigonometric functions and their inverses is based on the current trigonometric mode setting.

If an overflow results or an illegal operation is attempted, each of these routines will terminate with flag C set, and an error code in UH. Included are ERROR 37 (overflow), ERROR 38 (division by zero), and ERROR 39 (illegal function input argument).

R0+R2 into R0: Call F0 (EFBA)
R0-R2 into R0: Subroutine EFB6
R0*R2 into R0: Call 7E (F01A)
R0/R2 into R0: Call 58 (F084)
R0↑R2 into R0: Subroutine F89C (Register R6 used)

R0*R0 into R0: Subroutine F019
1/R0 into R0: Call 6E (F080). (XH and YH must contain 7A.)
SQR(R0) into R0: Subroutine F0E9

ABS(R0) into R0: Subroutine F597 (X, Y, and R2 will be unchanged.)
INT(R0)into R0: Subroutine F5BE
SGN(R0) into R0: Subroutine F59D (Y and R2 will be unchanged.)

Round R0 to ten digits: Subroutine F932 (Y and R2 will be unchanged.)

3.14159265359 (PI) into R0: Subroutine F5B5. (PI also stored into R2)

LN(R0) into R0: Subroutine F161
LOG(R0) into R0: Subroutine F165
EXP(R0) into R0: Subroutine F1CB
10↑R0 into R0: Subroutine F1D4

SIN(R0) into R0: Subroutine F3A2 (Register R6 used)
COS(R0) into R0: Subroutine F391 (Register R6 used)
TAN(R0) into R0: Subroutine F39E
ASN(R0) into R0: Subroutine F49A (Register R6 used)
ACS(R0) into R0: Subroutine F492 (Register R6 used)
ATN(R0) into R0: Subroutine F496 (Precede by clearing flag C)

DEG(R0) into R0: Subroutine F531 (Register R6 used)
DMS(R0) into R0: Subroutine F564 (Register R6 used)

COMPARISONS

Subroutine D0D2 performs a comparison of R0 and R2. The comparison performed will depend on the contents of A, as follows:

A=00:	R0<>R2	A=04:	R0=R2
A=01:	R0<R2	A=05:	R0<=R2
A=02:	R0>R2	A=06:	R0>=R2

If the test condition is met, R0 will contain 1, with flag Z clear; otherwise, R0 will contain zero, with flag Z set. Following execution, YH will not necessarily contain 7A.

CONSTANTS (FROM ROM TABLE) INTO R2

In each routine, XH and YH must contain 7A prior to execution.

1 into R2: Call 6A (F88F)
3.14159265359 (PI) into R2: Subroutine F875

57.2957795131 (180/PI) into R2: Subroutine F866
0.434294481903 (1/LN 10) into R2: Subroutine F87B
0.6 into R2: Call 62 (F88B)
0.9 into R2: Subroutine F87F
90 into R2: Subroutine F883
180 into R2: Subroutine F887

SEPARATION OF INTEGER AND FRACTION

Call 60 (F6B4) will separate the integral and fractional parts of R0, placing the integral part into R0 and the fractional part into R2. Neither result will be normalized. The given value of R0 is taken to be positive. XH and YH must contain 7A prior to execution.

REMOVAL OF SIGN

Call 6C (F6FB) loads the sign byte of R0 into A, and makes R0 positive. XH and YH must contain 7A prior to execution.

NORMALIZATION OF R0

Either of the subroutines Call E8 (F661) or Call 52 (F663) will normalize the contents of R0. The former applies a positive sign to the result; the latter takes the contents of A as the sign byte. In both routines, underflow results in zero; overflow sets flag C, with the code for ERROR 37 in Register UH.

Register XH must contain 7A prior to execution. Register Y is not changed by either routine.

BINARY-TO-DECIMAL CONVERSION

A two-byte binary number contained in Register U is converted to its decimal equivalent by execution of Call 10 (DD2D). The instruction calling this subroutine must be followed by a data byte, which specifies the effect produced. Two options exist: the converted value may be stored into R0; or, character codes for its display may be stored into memory, beginning at the address contained in Register Y.

Data byte	Interpretation of U	Disposition of result
00	unsigned binary	stored into R0
40	unsigned binary	characters stored, no sign
60	unsigned binary	characters stored, with sign
80	signed binary	stored into R0
C0	signed binary	characters stored, no sign
E0	signed binary	characters stored, with sign

Register YH will not necessarily contain 7A following execution.

DECIMAL-TO-BINARY CONVERSION

The binary equivalent of the contents of R0 (ignoring fractions) is calculated and stored into Register U by Call D0 (D0F9). The result is also stored into R0 (in binary format). A will equal UL.

Two data bytes must follow the instruction calling this subroutine.

The first of these specifies the range of values permitted; the second is added to the return address when the contents of R0 are not within that range, in which case UH will contain the code for ERROR 19.

Tabulated below are the permitted ranges (in decimal) for given values of the first passed byte. (When negative values are permitted, results are in signed binary.)

00:	0 to 65535	0A:	0 to 155
02:	0 to 65279	0C:	0 to 80
04:	-32768 to 32767	0E:	0 to 26
06:	0 to 32767	10:	0 to 25
08:	0 to 255		

If any of the above passed bytes is increased by 1, zero will be eliminated as a permitted value.

BINARY ROUTINES

STORAGE AND RECALL OF TWO-BYTE DATA

Call F6 (DDB5), followed by data bytes nn nn, stores UH into address nnnn and UL into nnnn+1. Following execution, A will contain UL; X will contain nnnn+1; Y and U are unchanged.

Call F4 (DBBC), followed by data bytes nn nn, loads UH with the contents of nnnn and UL with the contents of nnnn+1. A will contain the same byte as UH; X and Y are unchanged.

Call CA (C001), followed by data byte nn, stores XH into address 78nn and XL into 78nn+1. Following execution, A will contain XL; U will contain 78nn+1; X and Y are unchanged.

Call CC (DDC8), followed by data byte nn, loads XH with the contents of 78nn and XL with the contents of 78nn+1. A will contain the same byte as XH; Y and U are unchanged.

OPERATIONS IN BINARY

Subroutine DAA8 will replace U by its complement. (This is equivalent to replacing U by &10000-U.) X and Y are unchanged.

Subroutine DFE2 replaces U by U-X. If the result is negative, flag C is cleared, with UH containing 16. X and Y are unchanged.

Call 50 (DA71) replaces Y by U*Y; the result is also placed into X. If the result exceeds FFFF, flag C is set. Contents of U are lost.

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483